Albert-Ludwigs-Universität
Institut für Informatik
Prof. Dr. F. Kuhn

May 8, 2018

Network Algorithms, Summer Term 2018
Problem Set 1 – Sample Solution

Exercise 1: Vertex Coloring

1. Assume each node has degree Δ. In each round (including the first round) each undecided node sends Δ messages. Once a node becomes decided, it sends once more Δ messages. In each round, at least, but also in the worst case (e.g. the complete graph on n nodes), at most one node becomes decided. Therefore the total number of "undecided" messages and termination messages is in the worst case

$$\sum_{i=0}^{n}(n-i)\Delta + n\Delta = \Delta(\frac{n(n-1)}{2} + n) = \Theta(\Delta n^2).$$

Since on the clique $\Delta = n - 1$, we get that the total number of messages in the worst case is $\Theta(n^3)$.

2. First, note that all "undecided" messages can be replaced with sending no message. Initially nodes send their name to their neighbors. Then only local minima send a message, and even then only to those neighbors that are still undecided. Initially two messages get sent over each edge. During the execution of the algorithm, exactly one of the endpoints of an edge terminates first, and therefore sends a message over that edge. The total number of messages is therefore $3|E| = O(n^2)$.

Exercise 2: Coloring Rings and Trees

1. The log-star algorithm for the ring is basically identical to the algorithm for trees. Nodes do not have a parent in the ring, therefore we simply define the left neighbor of any node to be its "parent". Given this definition, we can run the normal log-star algorithm. Using the same argumentation as for trees, it can be shown that no two neighboring nodes choose the same color. Note that we can omit the "shift" step, as all "children" (i.e., the right neighbor) always have the same color.

2. We use two additional colors, ℓ and r, to solve the termination problem. Furthermore, we let each node send its color to both neighbors between each round of the log-star algorithm. This way, each node always knows the colors of both neighbors at the beginning of a round of the log-star algorithm (Algorithm 4).

   The algorithm works as follows for a node v: As long as neither v nor one of its neighbors has a color in $\mathcal{R} \cup \{\ell, r\}$, it executes Algorithm 4. If v learns that the color of its left neighbor is in $\mathcal{R}$ (regardless of the color of the right neighbor), and v's color is not in $\mathcal{R}$, then v recolors itself with the color ℓ and waits until both its neighbors have a color in $\mathcal{R} \cup \{\ell, r\}$. If a node v learns that the color of its right neighbor is in $\mathcal{R}$, while the color of its left neighbor and its own color are both not in $\mathcal{R}$, then v recolors itself with the color r and waits until both its neighbors have a color in $\mathcal{R} \cup \{\ell, r\}$. Additionally, as a node v to the right of a node colored ℓ no longer receives

---
Algorithm 1 Synchronous "3"-Coloring on Ring

1: send c_v to both neighbors
2: **while** $c_v \notin \mathcal{R} \cup \{\ell, r\}$ **do**
3:     **if** $c_\ell \in \mathcal{R}$ **then**
4:         $c_v := \ell$;
5:     **else if** $c_r \in \mathcal{R}$ **then**
6:         $c_v := r$;
7:     **else if** $c_\ell = \ell$ **then**
8:         do one step of Algorithm 4 (Lines 5–10) with an arbitrary color $c \in \mathcal{R}$ as parent color
9:     **else**
10:         do one step of Algorithm 4 (Lines 5–10)
11:     **end if**
12:     send c_v to both neighbors
13: **end while**
14: **wait until** both neighbors' colors are in $\mathcal{R} \cup \{\ell, r\}$
15: **while** $c_v \notin \{0, 1, 2\}$ **do**
16:     $x := (roundnumber \mod 5) + 3$;
17:     **if** $(c_v = x) \vee (c_v = \ell \wedge x = 6) \vee (c_v = r \wedge x = 7)$ **then**
18:         choose new color $c_v \in \{0, 1, 2\}$ with **First Free**
19:         send c_v to both neighbors
20:     **end if**
21: **end while**
---

new colors, we need the rule that v simply takes an arbitrary color $c \in \mathcal{R}$ as the new color of its parent and computes its new color based on c and its own color in each round (Line 8).

Inside the first while-loop, v eventually reaches a color in $\mathcal{R} \cup \{\ell, r\}$. After that, node v waits until its neighbors have also acquired a color in this range in order to start the color reduction phase. In each round of the color reduction phase (Lines 13–17), one color of $\{3, 4, 5, \ell, r\}$ is replaced by a legal color in $\{0, 1, 2\}$. As all nodes know the absolute number of rounds that have passed so far, this can be done without interference.

To prove correctness, we have to show that when reaching Line 13 of the algorithm no two neighboring nodes u and v have picked the same color in $\mathcal{R} \cup \{\ell, r\}$. We show this for ℓ in more detail:

Lemma. Two neighbors u and v cannot both be colored ℓ.

Proof. Let u be the parent of v w.l.o.g. A node only adopts ℓ if its color is not in $\mathcal{R}$, but its parent is in $\mathcal{R}$. This condition cannot hold for two neighboring nodes at the same time. Hence, u and v cannot reach color ℓ in the same round.

If u reaches color ℓ first, then v will reduce its color according to an arbitrary color from $\mathcal{R}$ in each following round and thus, it will never choose color ℓ.

If v reaches color ℓ first, u's color must be in $\mathcal{R}$ and thus u will not change its color again, in particular not to color ℓ, proving that two neighboring nodes cannot be colored ℓ. □

The same argumentation applies also to the color r. The logic of Algorithm 4 ensures that no two neighboring nodes get the same color $c \in \mathcal{R}$. Note that a node v to the right of a node colored ℓ acts like the root in a tree and can thus simply choose an arbitrary color $c \in \mathcal{R}$ in each round without causing any conflicts.

The running time of the algorithm is $1 + \log^* n + 5 \in O(\log^* n)$.